Caso Policía Nacional

El departamento de Inteligencia Militar que está asociado a la policía nacional está trabajando después de la pandemia con el doble de casos debido a un aumento en casos especiales de inteligencia. Por esta razón, se han visto atrasados en muchos casos ya que el personal con el que cuenta no es suficiente para analizar toda la data que ellos explotan. Debido a esto, les interesa automatizar el proceso de extracción de información importante y palabras claves de los miles documentos que reciben diarios por fotografías tomadas por espías.

Toda esta información la almacenan en una carpeta local secreta en donde cada imagen se nombra bajo la sintaxis "caso_sec_img_xx.png".

Lo que necesitan es un sistema que extraiga el texto de una imagen y luego analice su contenido para retener sus palabras claves y atributos principales.

● Recursos locales que emplearán.

-Utilizamos Computadora de escritorio.

-Utilizamos una laptop.

-También estuvimos utilizando programas en nuestros equipos locales como Visual studio code.

-Lenguaje de Programación: Python.

-Almacenamiento local para guardar imágenes y documentos.

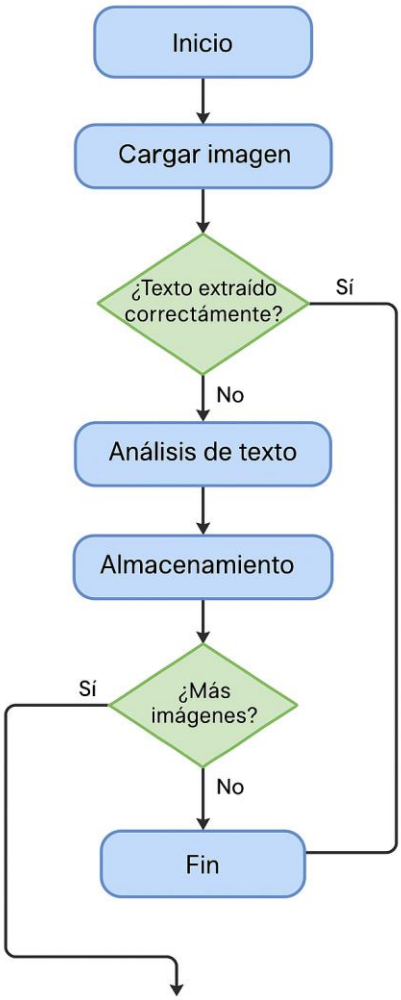
Recursos en la nube que usarán. Módulos de Azure ML que emplearán y por qué.

-Azure Computer Vision:
Este módulo nos ayuda a “leer” lo que dice una imagen. Es como si le enseñáramos a una computadora a ver una foto con texto (como una hoja escaneada o una nota tomada por un espía) y convertir todo eso en texto digital que luego se puede analizar.

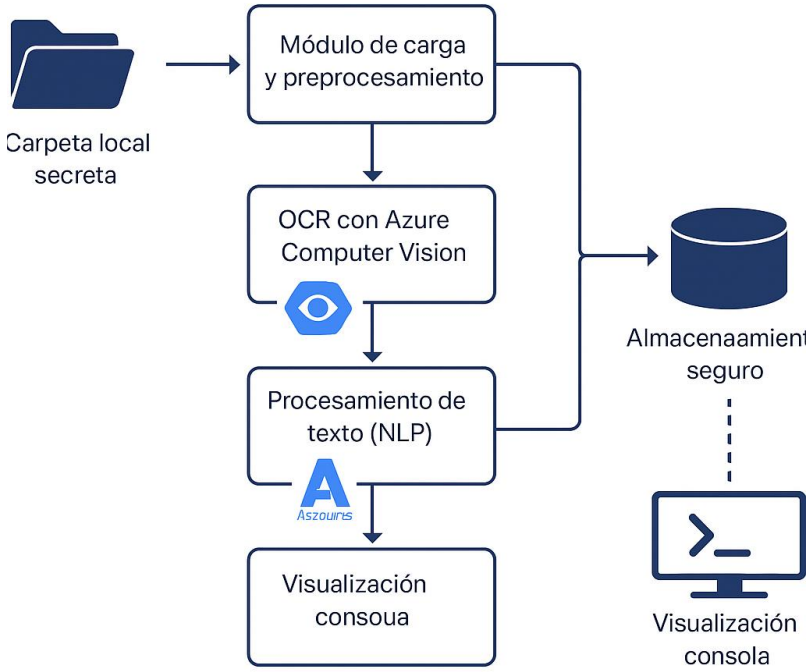
Azure Text Analytics (análisis de texto):
Una vez que ya tenemos el texto extraído, este otro módulo nos permite entender el contenido. Por ejemplo, detecta cuáles son las palabras más importantes, temas clave o entidades (como nombres de lugares, personas o cosas). Esto es súper útil porque ayuda a **identificar rápidamente de qué trata cada documento sin tener que leerlo completo.**

tkinter (herramienta visual en Python):
Como parte del sistema, también agregamos una ventanita emergente que muestra el texto que se extrajo de la imagen. Esto se hace con tkinter, que es una librería en Python que permite crear interfaces gráficas sencillas. Así, cuando el sistema termina de procesar una imagen, le enseña al usuario el resultado directamente en pantalla, de forma rápida y fácil de ver.

● Diagrama de flujo de los datos y proceso.



Infraestructura del sistema que propone como Solucion



● Procedimiento realizado para resolver el problema.

Primero usamos una herramienta que "lee" las imágenes

Para comenzar, usamos una tecnología llamada OCR (Reconocimiento Óptico de Caracteres). En nuestro caso, utilizamos herramientas de Microsoft Azure como Computer Vision y Text Analytics, que básicamente hacen que la computadora pueda mirar una imagen y leer lo que dice como si fuera un ser humano.

Por ejemplo, si hay una foto de un documento secreto, esta herramienta puede convertir ese texto en algo digital que se pueda copiar y analizar.

Después, analizamos ese texto con inteligencia artificial

Una vez que ya tenemos el texto, lo pasamos por un modelo de lenguaje, que es como un robot inteligente que sabe entender lo que dice ese texto.

Este modelo busca automáticamente las palabras más importantes, los nombres, lugares, fechas o cualquier información clave que pueda ayudar al departamento de inteligencia a resolver los casos más rápido.

Lo probamos con imágenes reales
Para asegurarnos de que todo funciona bien, hicimos pruebas usando imágenes reales, como las que recibiría el departamento en su trabajo diario. Esto nos ayudó a comprobar que el sistema puede entender distintos tipos de letra, documentos borrosos o con mala calidad, y aún así extraer bien la información.

● Método de recolección de datos y recopilación de la información de entrenamiento.

Usamos la misma imagen de nuestro proyecto, relacionada con el caso de la Policía Nacional
Para comenzar, probamos el sistema con una imagen que nosotros mismos preparamos, simulando un documento real que podría recibir el departamento de Inteligencia Militar. Esta imagen nos ayudó a ver cómo el sistema responde a un caso controlado, hecho específicamente para este proyecto.

También usamos otra imagen sacada de internet, relacionada con casos policiales reales
Además, buscamos una imagen diferente en Google, que también representaba un caso policial, pero de una fuente externa. Esto lo hicimos para ver si el sistema podía adaptarse a otros estilos de documentos, diferentes tipos de letra, calidades de imagen o formatos.
Así nos aseguramos de que el sistema funcione bien con información real del mundo exterior, no solo con lo que nosotros le damos.

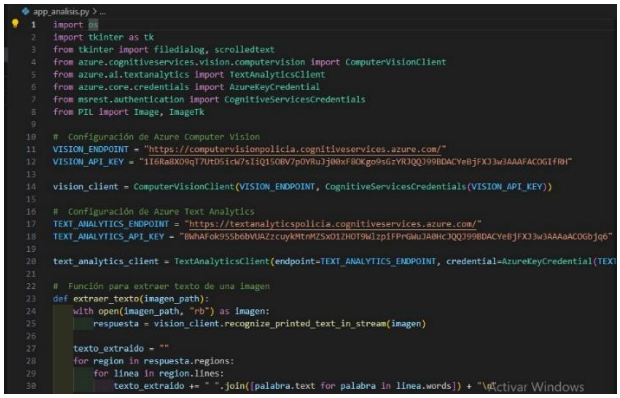
Formato de salida propuesto de la información.

El formato de salida seria por Consola o Terminal Para que el sistema sea lo más práctico posible, decidimos que la información procesada se muestre directamente en la consola o terminal del programa, es decir, en texto plano y simple.

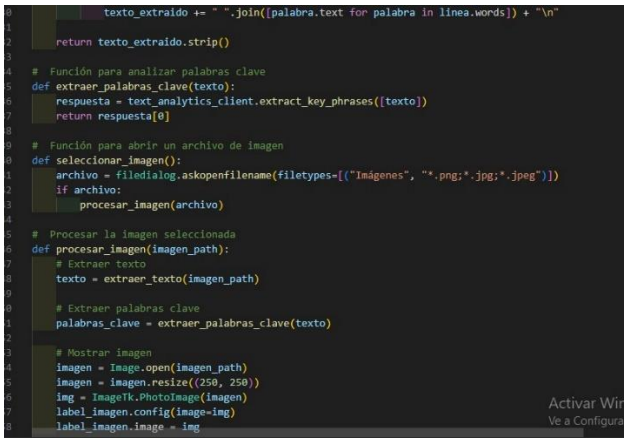
Es rápido y directo.

No necesita instalar programas extras.

Es ideal para pruebas, para ver si el sistema está funcionando correctamente.



```python
import os
import tkinter as tk
from tkinter import filedialog, scrolledtext
from azure.cognitiveservices.vision.computervision import ComputerVisionClient
from azure.ai.textanalytics import TextAnalyticsClient
from azure.core.credentials import AzureKeyCredential
from msrest.authentication import CognitiveServicesCredentials
from PIL import Image, ImageTk

#  Configuración de Azure Computer Vision
VISION_ENDPOINT = "https://computervisionpolicia.cognitiveservices.azure.com/"
VISION_API_KEY = "1IGRaBXO9qT7UtD5ickV7sI1Q1SOBV7pOYRuJj00xF8OKgo9sGzYNJQQJ99BDACYeBjFXJ3w3AAAFACOGIfNH"

vision_client = ComputerVisionClient(VISION_ENDPOINT, CognitiveServicesCredentials(VISION_API_KEY))

#  Configuración de Azure Text Analytics
TEXT_ANALYTICS_ENDPOINT = "https://textanalyticspolicia.cognitiveservices.azure.com/"
TEXT_ANALYTICS_API_KEY = "BWhAFok9S5b6bVUAZzcuykMtnMZ5xOIZHOT9Wlzp1FPrGWuJA0HcJQQJ99BDACYeBjFXJ3w3AAAaACOGbjQ6"

text_analytics_client = TextAnalyticsClient(endpoint=TEXT_ANALYTICS_ENDPOINT, credential=AzureKeyCredential(TEXT

#  Función para extraer texto de una imagen
def extraer_texto(imagen_path):
    with open(imagen_path, "rb") as imagen:
        respuesta = vision_client.recognize_printed_text_in_stream(imagen)

    texto_extraido = ""
    for region in respuesta.regions:
        for linea in region.lines:
            texto_extraido += " ".join([palabra.text for palabra in linea.words]) + "\n"
```

```python
            texto_extraido += " ".join([palabra.text for palabra in linea.words]) + "\n"

    return texto_extraido.strip()

#  Función para analizar palabras clave
def extraer_palabras_clave(texto):
    respuesta = text_analytics_client.extract_key_phrases([texto])
    return respuesta[0]

#  Función para abrir un archivo de imagen
def seleccionar_imagen():
    archivo = filedialog.askopenfilename(filetypes=[("Imágenes", "*.png;*.jpg;*.jpeg")])
    if archivo:
        procesar_imagen(archivo)

#  Procesar la imagen seleccionada
def procesar_imagen(imagen_path):
    # Extraer texto
    texto = extraer_texto(imagen_path)

    # Extraer palabras clave
    palabras_clave = extraer_palabras_clave(texto)

    # Mostrar imagen
    imagen = Image.open(imagen_path)
    imagen = imagen.resize((250, 250))
    img = ImageTk.PhotoImage(imagen)
    label_imagen.config(image=img)
    label_imagen.image = img
```


```python
def procesar_imagen(imagen_path):
    imagen = imagen.resize((250, 250))
    img = ImageTk.PhotoImage(imagen)
    label_imagen.config(image=img)
    label_imagen.image = img

    # Mostrar resultados en la interfaz
    text_resultado.delete(1.0, tk.END)
    text_resultado.insert(tk.END, f" Texto Extraído:\n{texto}\n\n")
    text_resultado.insert(tk.END, f" Palabras Clave: {', '.join(palabras_clave)}")

#  Crear la interfaz gráfica
root = tk.Tk()
root.title("Análisis de Imágenes con IA ")
root.geometry("600x500")

frame = tk.Frame(root)
frame.pack(pady=10)

btn_seleccionar = tk.Button(frame, text=" Seleccionar Imagen", command=seleccionar_imagen, font=("Arial", 12))
btn_seleccionar.pack()

label_imagen = tk.Label(root)
label_imagen.pack(pady=10)

text_resultado = scrolledtext.ScrolledText(root, width=70, height=10, font=("Arial", 10))
text_resultado.pack(pady=10)

# Ejecutar la interfaz gráfica
root.mainloop()
```


- **Rol de los miembros del equipo y trabajo realizado.**

Yoelvy – Programación del análisis de texto:
A Yoelvy le toca la parte de escribir el código que se encarga de analizar el texto que se extrae de las imágenes. Básicamente, él va a hacer que el sistema entienda lo que dice el texto, identificando las palabras más importantes, organizando la información y preparándola para que sea útil. Es como enseñarle al sistema a leer y a sacar lo más relevante de lo que lee.

Wandy – Configuración del OCR y pruebas con imágenes:
Wandy se está encargando de poner a punto el OCR, que es la herramienta que "lee" el texto en las imágenes. Él va a probar con diferentes tipos de imágenes (claras, borrosas, con diferentes fondos, etc.) para asegurarse de que el sistema pueda reconocer bien las letras sin importar cómo se vea la imagen. En resumen, él está afinando la vista del sistema para que no se le escape nada.

● Costos de la propuesta.

Microsoft Azure Estimate

Su presupuesto

Service category	Service type	Custom name	Region	Description	Estimated monthly cost	Estimated upfront cost
IA y Machine Learning	Azure AI services		East US	Lenguaje de Azure AI, Pago por uso, Estándar, AI Services para lenguaje: 2 x 1000 registros de texto, Resumen: 20 x 1 000 registros de texto, Text Analytics de mantenimiento: 5 x 1 000 registros de texto, Respuesta a preguntas personalizada y respuesta a preguntas predefinidas: 4 x 1000 registros de texto, Reconocimiento de entidades con nombre personalizado y clasificación de texto personalizado: 4 x 1000 registros de texto, 8 horas de entrenamiento 8 modelos de hospedaje de puntos de conexión, Flujo de trabajo de orquestación y reconocimiento de lenguaje conversacional: 5 x 1000 registros de texto, 8 Horas de aprendizaje avanzado	$130.00	$0.00
Support			Support		$100.00	$0.00
			Licensing Program	Microsoft Customer Agreement (MCA)		
			Billing Account			
			Billing Profile			
			Total		$230.00	$0.00

Disclaimer

All prices shown are in United States – Dollar ($) USD. This is a summary estimate, not a quote. For up to date pricing information please visit https://azure.microsoft.com/pricing/calculator/
This estimate was created at 4/5/2025 12:11:22 AM UTC.

● Métricas de efectividad del sistema propuesto.

Precisión del OCR	Porcentaje de caracteres correctamente identificados en comparación con el texto original.	≥ 95%
Tasa de Extracción de Texto Válido	Porcentaje de imágenes en las que se logró extraer texto útil.	≥ 90%
Exactitud en Identificación de Palabras Clave	Porcentaje de palabras clave verdaderamente relevantes que el sistema logra extraer.	≥ 85%

Tasa de Falsos Positivos	Palabras marcadas como clave pero que no lo son.	≤ 15%
Tasa de Falsos Negativos	Palabras clave que el sistema no detectó.	≤ 15%
Tasa de Éxito en Imágenes con Calidad Baja	Porcentaje de éxito en la extracción de texto en imágenes con baja calidad.	≥ 50%
Compatibilidad con Formatos Diversos	Capacidad para procesar imágenes en distintos formatos como .png, .jpg, etc.	Al menos 2 formatos comunes

● Guia de uso de la demo realizada.

-Vamos al apartado Analizar_Texto.

-Adjuntamos la imagen de nuestra carpeta secreta.

-Luego que nos arroje el texto de la imagen lo copiamos.

-Vamos al apartado de Analizar texto para que nos arroje las palabras claves.

● Dificultades al realizar el proyecto.

-Reconocimiento de texto en imágenes de baja calidad

-Selección precisa de palabras clave

-Organización y pruebas con imágenes simuladas

● Posibles mejoras de la propuesta.

-Hacer que reconozca también texto escrito a mano.

- Manejo de errores → Si la API falla, muestra un mensaje claro.

-Verificación de credenciales → Si hay un problema con la clave o endpoint, avisa.

-Un entorno visual más atractivo.

Riesgos y limitaciones.

-Si la imagen está muy borrosa, el OCR falla.

-Si el espía usa códigos o lenguaje oculto, la IA puede no entender.

-Hay riesgo de que alguien acceda a la carpeta secreta si no está bien protegida.